PORTAL Subscribe Register Login
(Full Service) (Limited Service, Free)

Search: ● The ACM Digital Library ○ The Guide
compiler-directed instruction marking or prefix cache write-through <and>

THE ACM DIGITAL LIBRARY Feedback

Terms used
compiler directed instruction marking or prefix cache write through and reduction cache miss and code in

Sort results by relevance
Display results expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try
Try

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9
Best 200 shown

1 Examination of a memory access classification scheme for pointer-intensiv
Luddy Harrison
January 1996 Proceedings of the 10th international conference on Supercom
Full text available: pdf(991.11 KB) Additional Information: full citation, references, citing

Keywords: CPU architecture, data cache, instruction profiling, memory access tolerance

2 A general framework for prefetch scheduling in linked data structures and i prefetching
Seungryul Choi, Nicholas Kohout, Sumit Pamnani, Dongkeun Kim, Donald Yeung
May 2004 ACM Transactions on Computer Systems (TOCS), Volume 22 Issu
Full text available: pdf(2.45 MB) Additional Information: full citation, abstract, reference

Pointer-chasing applications tend to traverse composite data structures consis chains. While the traversal of any single pointer chain leads to the serializatio independent pointer chains provides a source of memory parallelism. This arti *interchain memory parallelism* for the purpose of memory latency tolerance, u *prefetching*. Previous work ...

Keywords: Data prefetching, memory parallelism, pointer-chasing code

3 Speeding up irregular applications in shared-memory multiprocessors: mer
Zheng Zhang, Josep Torrellas
May 1995 ACM SIGARCH Computer Architecture News , Proceedings of the 22nd a
Computer architecture, Volume 23 Issue 2
Full text available: pdf(1.74 MB) Additional Information: full citation, abstract, references, ci

While many parallel applications exhibit good spatial locality, other important
problem-solving or CAD do not. Often, these irregular codes contain small rec
Consequently, while the former applications benefit from long cache lines, the
solution is to combine short lines with prefetching. In this way, each applicati
locality that it has. However, prefetching, if provided, ...

4 Compiler scheduling: Compiler managed micro-cache bypassing for high p
Youfeng Wu, Ryan Rakvic, Li-Ling Chen, Chyi-Chang Miao, George Chrysos, Jess
November 2002 Proceedings of the 35th annual ACM/IEEE international symposi
Full text available: pdf(1.15 MB) Publisher Site Additional Information: full citation, abs

Advanced microprocessors have been increasing clock rates, well beyond the
performance microprocessors, a small and fast data micro cache (ucache) is ir
proper management of it via load bypassing has a significant performance im
evaluate a hardware-software collaborative technique to manage ucache bypa
supports the ucache bypassing with a flag in the load ...

5 Instruction prefetching of systems codes with layout optimized for reduced
Chun Xia, Josep Torrellas
May 1996 ACM SIGARCH Computer Architecture News , Proceedings of the 23rd a
Computer architecture, Volume 24 Issue 2
Full text available: pdf(1.65 MB) Additional Information: full citation, abstract, references, ci

High-performing on-chip instruction caches are crucial to keep fast processors
caches are usually successful at intercepting instruction fetches in loop-intens
able to do so in large systems codes. To improve the performance of the latte
out the code in memory for reduced cache conflicts. Interestingly, such an op
can be exploited by a new type of ...

6 Memory and network optimization in embedded designs: Multi-profile base
E. Wanderley Netto, R. Azevedo, P. Centoducatte, G. Araujo
June 2004 Proceedings of the 41st annual conference on Design automation
Full text available: pdf(272.41 KB) Additional Information: full citation, abstract, referen

Code compression has been shown to be an effective technique to reduce cod
systems. It has also been used as a way to increase cache hit ratio, thus redu
performance. This paper proposes an approach to mix static/dynamic instructi
so as to best exploit trade-offs in compression ratio/performance. Compressec
indices into fixed-size codewords, el ...

Keywords: code compression, code density, compression

7 Reducing cache misses using hardware and software page placement
Timothy Sherwood, Brad Calder, Joel Emer
May 1999 Proceedings of the 13th international conference on Supercomputing
Full text available: pdf(1.50 MB) Additional Information: full citation, references, citings, index terms

8 Cache-conscious data placement
Brad Calder, Chandra Krintz, Simmi John, Todd Austin
October 1998 Proceedings of the eighth international conference on Architectural :
operating systems, Volume 33 , 32 Issue 11 , 5
Full text available: pdf(1.49 MB) Additional Information: full citation, abstract, references, ci

As the gap between memory and processor speeds continues to widen, cache
component of processor performance. Compiler techniques have been used to
by mapping code with temporal locality to different cache blocks in the virtual
conflicts. These code placement techniques can be applied directly to the prob
cache pedormance.In this paper we present a gene ...

9 Compiler and hardware support for cache coherence in large-scale multipr
performance study
Lynn Choi, Pen-Chung Yew
May 1996 ACM SIGARCH Computer Architecture News , Proceedings of the 23rd a
Computer architecture, Volume 24 Issue 2
Full text available: pdf(1.48 MB) Additional Information: full citation, abstract, reference

In this paper, we study a hardware-supported, compiler directed (HSCD) cach
implemented on a large-scale multiprocessor using off-the-shelf microprocess
adapted to various cache organizations, including multi-word cache lines and
system related issues, including critical sections, inter-thread communication,
addressed. The cost of the required hardware sup ...

10 Physical Experimentation with Prefetching Helper Threads on Intel's Hyper
Dongkeun Kim, Steve Shih-wei Liao, Perry H. Wang, Juan del Cuvillo, Xinmin Ti
Yeung, Milind Girkar, John P. Shen
March 2004 Proceedings of the international symposium on Code generation and
runtime optimization
Full text available: pdf(264.47 KB) Additional Information: full citation, abstr

Pre-execution techniques have received much attention as aneffective way of
ever-increasingmemory latency. A number of pre-execution techniquesbased
been proposed andstudied extensively by researchers. They report promising
Simultaneous Multithreading (SMT)processor. In this paper, we apply the help
multithreaded machine, i.e., Intel Pentium 4 processor withHyp ...

11 Compiler-directed run-time monitoring of program data access
Chen Ding, Yutao Zhong
June 2002 ACM SIGPLAN Notices , Proceedings of the workshop on Memory systei supplement
Full text available: pdf(1.40 MB) Additional Information: full citation, abstract, referer

Accurate run-time analysis has been expensive for complex programs, in part data. Some applications require only partial reorganization. An example of thi: from a mobile device. Complete monitoring is not necessary because not all a support partial monitoring, this paper presents a framework that includes a sc run-time monitor. The compiler inserts ru ...

12 Trace-driven memory simulation: a survey
Richard A. Uhlig, Trevor N. Mudge
June 1997 ACM Computing Surveys (CSUR), Volume 29 Issue 2
Full text available: pdf(636.11 KB) Additional Information: full citation, abstract, references, citin

As the gap between processor and memory speeds continues to widen, metho designs before they are implemented in hardware are becoming increasingly i trace-driven memory simulation, has been the subject of intense interest amc enjoyed rapid development and substantial improvements during the past dec these developments by establishing criteria for evaluating trac ...

Keywords: TLBs, caches, memory management, memory simulation, trace-dri

13 Using generational garbage collection to implement cache-conscious data |
Trishul M. Chilimbi, James R. Larus
October 1998 ACM SIGPLAN Notices , Proceedings of the first international sympo: 34 Issue 3
Full text available: pdf(1.20 MB) Additional Information: full citation, abstract, references, ci

The cost of accessing main memory is increasing. Machine designers have trie processor and memory technology trends underlying this increasing gap with . tolerate memory latency. These techniques, unfortunately, are only occasiona programs. Recent research has demonstrated the value of a complementary a structures are reorganized to improve cache loca ...

Keywords: cache-conscious data placement, garbage collection, object-oriente

14 Cache coherence in large-scale shared-memory multiprocessors: issues ai
David J. Lilja
September 1993 ACM Computing Surveys (CSUR), Volume 25 Issue 3
Full text available: pdf(3.12 MB) Additional Information: full citation, references, citings, index terr

15 Memory data organization for improved cache performance in embedded p
Preeti Ranjan Panda, Nikil D. Dutt, Alexandru Nicolau
October 1997 ACM Transactions on Design Automation of Electronic Systems (TC
Full text available: pdf(534.55 KB) Additional Information: full citation, abstract, references,

Code generation for embedded processors opens up the possibility for several that have been ignored by traditional compilers due to compilation time const into account the parameters of the data caches for organizing scalar and array into memory, with the objective of improving data cache performance. We pre to minimize compulsory cache misse ...

Keywords: cache memory, data cache, memory synthesis, system design, sys

16 Cache-conscious structure definition
Trishul M. Chilimbi, Bob Davidson, James R. Larus
May 1999 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 confere and implementation, Volume 34 Issue 5
Full text available: pdf(1.30 MB) Additional Information: full citation, abstract, references, ci

A program's cache performance can be improved by changing the organizatior pointer-based data structures. Previous techniques improved the cache perfor distinct instances to increase reference locality. These techniques produced si but worked best for small structures that could be packed into a cache block.T concentrating on the internal organization of f ...

Keywords: cache-conscious definition, class splitting, field reorganization, stru

17 Compiler transformations for high-performance computing
David F. Bacon, Susan L. Graham, Oliver J. Sharp
December 1994 ACM Computing Surveys (CSUR), Volume 26 Issue 4
Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citing

In the last three decades a large number of compiler transformations for optir implemented. Most optimizations for uniprocessors reduce the number of instr transformations based on the analysis of scalar quantities and data-flow techr high-performance superscalar, vector, and parallel processors maximize parall transformations that rely on tracking the properties o ...

Keywords: compilation, dependence analysis, locality, multiprocessors, optimi processors, vectorization

18 Predictability of load/store instruction latencies
Santosh G. Abraham, Rabin A. Sugumar, Daniel Windheiser, B. R. Rau, Rajiv Gu
December 1993 Proceedings of the 26th annual international symposium on Micro
Full text available: pdf(1.51 MB) Additional Information: full citation, references, citings

19 Memory system performance of programs with intensive heap allocation
Amer Diwan, David Tarditi, Eliot Moss
August 1995 ACM Transactions on Computer Systems (TOCS), Volume 13 Is
Full text available: pdf(2.10 MB) Additional Information: full citation, abstract, references, ci

Heap allocation with copying garbage collection is a general storage managem
languages. It is believed to have poor memory system performance. To invest
study of the memory system performance of heap allocation for memory syste
studied the performance of mostly functional Standard ML programs which ma
found that most machines support heap allocation poorly. Howeve ...

Keywords: automatic storage reclamation, copying garbage collection, garbag
collection, heap allocation, page mode, subblock placement, write through, wr
write-policy

20 The Wisconsin Wind Tunnel: virtual prototyping of parallel computers
Steven K. Reinhardt, Mark D. Hill, James R. Larus, Alvin R. Lebeck, James C. Le
June 1993 ACM SIGMETRICS Performance Evaluation Review , Proceedings of the
Measurement and modeling of computer systems, Volume 21 Issue 1
Full text available: pdf(1.40 MB) Additional Information: full citation, references, citings,

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7

CiteSeer Find: compiler-directed instruction marking Documents Citations

Searching for PHRASE **compiler directed instruction marking cache optimization**.
Restrict to: Header Title Order by: Expected citations Hubs Usage Date Try: Google (CiteSeer) Google (Web) CSB DBLP
No documents match Boolean query. Trying non-Boolean relevance query.
500 documents found. **Order: relevance to query.**

Optimizing Instruction Cache Performance for Operating.. - Torrellas, Xia, Daigle (1995) (Correct) (36 citations)
the hit rate of **caches** is to use an optimizing **compiler** to minimize **cache** interference via an improved
C code. We run 4 compilations, each on a **direct**ory of 22 C files. The file size is about 60 lines
Optimizing **Instruction Cache** Performance for Operating System
ftp.cs.uiuc.edu/pub/research-groups/csrd/iacoma/osplace.ps

Wrong-Path Instruction Prefetching - Jim Pierce (1994) (Correct) (25 citations)
Trace Description Instr Misses Traffic gcc Gnu C **compiler** 126M 111 16 gs PS file viewer 86M 110 16
instruction targets regardless of the predicted **direct**ion of conditional branches. The algorithm
Wrong-Path **Instruction** Prefetching Jim Pierce Trevor Mudge Intel
www.ece.orst.edu/~sllu/memory/Pierce.micro29.ps

Efficient Implementation of Software Release.. - Niwa, Inagaki.. (1997) (Correct)
distributed shared memory, release consistency, **compiler**, pointer-analysis ABSTRACT The shared memory
Via Anonymous Ftp From Ftp.is.s.u-Tokyo.ac.jp **direct**ory /pub/tech-Reports)Supplementary Notes Report
code reading to shared memory into a single load **instruction** and translates the code writing to shared
www.is.s.u-tokyo.ac.jp/tech-reports/TR97-05-a4.ps.gz

A Comprehensive Instruction Fetch Mechanism for a Processor.. - Yeh, Patt (1992) (Correct) (23 citations)
performance. Decreased latency means the newly **directed instruction** stream can begin execution faster.
1 -4, 1992, Portland, Oregon. A Comprehensive **Instruction** Fetch Mechanism for a Processor Supporting
www.eecs.umich.edu/HPS/pub/micro-92.instr-fetch.ps

Techniques for Compiler-Directed Cache Coherence - Choi, Lim, Yew (1996) (Correct)
Techniques for **Compiler-Directed Cache** Coherence Lynn Choi Intel
Techniques for **Compiler-Directed Cache** Coherence Lynn Choi Intel Corporation
processor. The RP3 also provides **cache** control **instructions** to invalidate the **cache** contents. The
www.cs.umn.edu/Research/Agassiz/Paper/choi.ieeepdt.ps.Z

An Accurate Instruction Cache Analysis Technique for.. - Lim, Min, Lee, Park, .. (1994) (Correct) (3 citations)
that the **instruction cache** has two blocks and is **direct**-mapped. In a **direct** mapped **cache**, each
Real-time Applications, April 1994. An Accurate **Instruction Cache** Analysis Technique for Real-time
Applications, April 1994. An Accurate **Instruction Cache** Analysis Technique for Real-time Systems
archi.snu.ac.kr/PUBLICATIONS/papers/real-time/sslim-wart-1994.ps.gz

Optimization of Machine Descriptions for Efficient Use - Gyllenhaal, Hwu, Rau (1996) (Correct) (14 citations)
Abstract A machine description facility allows **compiler** writers to specify machine execution constraints
of the machine's description (commonly coded **direct**ly into the **compiler**)that must be tediously
to the **optimization** and scheduling phases of an **instruction**-level parallelism (ILP) optimizing **compiler**.
ftp.crhc.uiuc.edu/pub/IMPACT/conference/micro-96-optimization.ps

Design and Performance Evaluation of a Cache Assist to implement.. - John (1997) (Correct) (6 citations)
but the processor can access annex **cache** entries **directly**, i. e. annex **cache** entries can bypass the main
Florida Tampa, FL 33620 Abstract Efficient **instruction** and data **caches** are extremely important for
Design and Performance Evaluation of a **Cache** Assist to implement Selective Caching L. John
www.ece.utexas.edu/~ljohn/annex.ps

Solving Graph Problems With Dynamic Computation Structures - Babb, Frank, Agarwal (1996) (Correct) (20 citations)
computing fabric. Thus, a virtual wires **compiler**, coupled with front-end commercial behavioral
for reconfigurable computing. DCS specializes **directed** graph instances into user-level hardware for
of dynamic code, 6 where new processor **instructions** are dynamically generated based on the input
ftp.cag.lcs.mit.edu/pub/raw/documents/Babb:SPIE:1996.ps.Z

Profile-Driven Instruction Level Parallel Scheduling with.. - Chekuri Dept (1996) (Correct) (10 citations)
level parallelism (ILP) is a critical problem in **compiler optimization** research, in light of the increased

graph of a single basic block will be a **directed** acyclic graph (DAG) 2]and in practice,
Profile-Driven **Instruction** Level Parallel Scheduling with Application to
theory.stanford.edu/~chekuri/postscript/micro96.ps.gz

Functional languages and very fine grained parallelism.. - Jon Mountjoy (1994) (Correct) (5 citations)
Abstract A functional language **compiler** can be used as a powerful tool in the scheduling
want to tackle this problem is that of **compiler directed instruction** level parallelism. The price to be
of programs for hardware capable of fine grained **instruction** level parallelism. There have been many
www.wins.uva.nl/pub/functional/reports/fine_grained_parallelism.ps.Z

Stack-Based Typed Assembly Language - Morrisett, Crary, Walker, Glew (1998) (Correct) (33 citations)
expressive to serve as a target language for **compilers** of high-level languages such as ML. That work
their dynamically typed counterparts. Modern type-**directed compilers** [18, 25, 7, 32, 19, 29, 11] exploit
generation including register allocation and **instruction** scheduling are left unchecked and types cannot
reports-archive.adm.cs.cmu.edu/anon/1998/CMU-CS-98-178.ps

Integrating Fine-Grained Message Passing In Cache Coherent.. - Poulsen, Yew (1996) (Correct) (3 citations)
prefetching, data forwarding, message passing, **compiler** algorithms, **cache** coherent shared memory
model a shared memory architecture with **direct**ory-based **cache** coherence and are driven by
application characteristics, to reduce processor **instruction** overheads, **cache** miss ratios, and memory
ftp.csrd.uiuc.edu/pub/misc/poulsen/jpdc95.final.ps.gz

Removing Interference Misses using Cache Bypass Buffers - Juan, Navarro, Lang (1994) (Correct)
passes. The choice of memory is done by the **compiler** with a simple analysis of the locality
by replacing the **cache** by a local memory, managed **direct**ly by the program. The first solution is
drawbacks: ffl The program has to include **instructions** to move data between main memory and the local
ftp.ac.upc.es/pub/reports/CEPBA/1994/UPC-CEPBA-94-14.ps.Z

The Jalapeño Dynamic Optimizing Compiler for Java - Burke, Choi, Fink.. (1999) (Correct) (43 citations)
The Jalape~no Dynamic Optimizing **Compiler** for Java TM Michael G. Burke Jong-Deok Choi
www.mcs.newpaltz.edu/~hind/papers/grande99.ps

SCHEME->C: a Portable Scheme-to-C Compiler - Bartlett (1989) (Correct) (2 citations)
Report 89/1 SCHEME-C a Portable Scheme-to-C **Compiler** Joel F. Bartlett d i g i t a l Western Research
Cambridge, Massachusetts (CRL)Our research is **directed** towards mainstream high-performance computer
in the Subject line you will receive detailed **instructions**. SCHEME-C a Portable Scheme-to-C **Compiler**
ftp.cs.indiana.edu/pub/scheme-repository/doc/pubs/s2c.ps.gz

A Quantitative Analysis of Instruction Prefetching - Kim, Min, Kim (1995) (Correct)
this table to make the prefetch decision in the **direct**ion with the highest frequency for each possible
301-307, Sep. 1995. A Quantitative Analysis of **Instruction** Prefetching S. B. Kim S. L. Min C. S. Kim
on misses prefetches the next block only on a **cache** miss. Tagged prefetch, an enhancement of
net.snu.ac.kr/archi/PUBLICATIONS/papers/cpu-cache/sbkim-euro-1995.ps.gz

Instruction Cache Fetch Policies for Speculative Execution - Lee, Baer, Calder, Grunwald (1995) (Correct) (12 citations)
of manual pages given to Groff. 39 17.5 gcc GNU C **Compiler**, version 1.35. The measurements show only the
uses a two-bit saturating counter to predict the **direct**ion of a conditional branch. In this BTB
Instruction Cache Fetch Policies for Speculative
www.cse.ucsd.edu/~calder/papers/ISCA-95-Spec.ps.Z

Designing Programming Languages for Analyzability: A Fresh.. - Hendren, Gao (1992) (Correct) (17 citations)
a programming language mechanism and associated **compiler** techniques which significantly enhance the
Thus, we treat the inductive specification as a **direct**ive, rather than a type. The programming language
with today's RISC processors, some degree of **instruction**-level parallelism is required to fully utilize
ftp.capsl.udel.edu/pub/doc/acaps/papers/ICCL92.ps.gz

Computer Design Strategy for MCM-D/Flip-Chip Technology - Franzon, al. (Correct)
eight rows of signal pins can break out in one **direct**ion (only four rows and the top layer of routing
to build a `MegaChip' CPU consisting of an **Instruction** Fetch Unit and Execution Unit. By building
Thus there is tremendous advantage to building the **caches** in a computer in an SRAM process and using an MCM
www.ece.ncsu.edu/info.ece/vlsi_info/techreports/NCSU-ERL-96-03.PS.Z

Documents 21 to 40 Previous 20 Next 20

Try your query at: Google (CiteSeer) Google (Web) CSB DBLP